Online Payments Fraud Detection

with Machine Learning

Annamacharya Institute of Technology & Sciences

Full Stack Development - Project Documentation

February 2026

Team ID: LTVIP2026TMIDS41562

P Varshitha	Team Leader
G Lokanath Reddy	Team Member
K Pujith	Team Member
T Sruthi	Team Member

Table of Contents

1. Introduction

The introduction of online payment systems has revolutionized financial transactions, making payments faster and more convenient. However, this digital transformation has also led to a significant increase in payment fraud. Online payment fraud can affect anyone using digital payment systems, particularly those involving credit cards, bank transfers, and mobile wallets.

This project addresses this critical challenge by building a Machine Learning-powered web application that detects fraudulent online payment transactions in real time. The system uses a trained Decision Tree Classifier to analyze transaction features and classify them as either "Fraud" or "No Fraud", providing instant fraud assessment.

The application is built using Python Flask for the backend, with a modern, responsive web interface for user interaction. The ML model is trained on the PaySim synthetic financial dataset, which simulates mobile money transactions based on real-world patterns.

2. Project Overview

2.1 Purpose

To develop a web-based fraud detection system that classifies online payment transactions as fraudulent or legitimate using machine learning, providing real-time fraud assessment to help financial institutions and end-users prevent unauthorized transactions.

2.2 Key Features

- Real-Time Fraud Prediction: Users input transaction details and receive instant classification.
- ML Model: Trained Decision Tree Classifier achieving 97.6% accuracy on test data.
- Premium Web Interface: Modern dark-themed UI with glassmorphism, animations, and responsive design.
- Multiple Transaction Types: Supports Cash Out, Payment, Cash In, Transfer, and Debit.
- Visual Risk Assessment: Risk level indicator (High/Low) displayed with prediction results.
- Statistics Dashboard: Shows model accuracy, response time, and feature count.

2.3 Technologies Used

Component	Technology	Version
Backend	Python Flask	3.x
ML Library	Scikit-Learn	1.x
Data Processing	NumPy / Pandas	Latest
Visualization	Matplotlib / Seaborn	Latest
Frontend	HTML5 / CSS3 / JavaScript	-
Templating	Jinja2 (Flask built-in)	-
Model Storage	Pickle	Built-in

3. System Architecture

3.1 Architecture Diagram

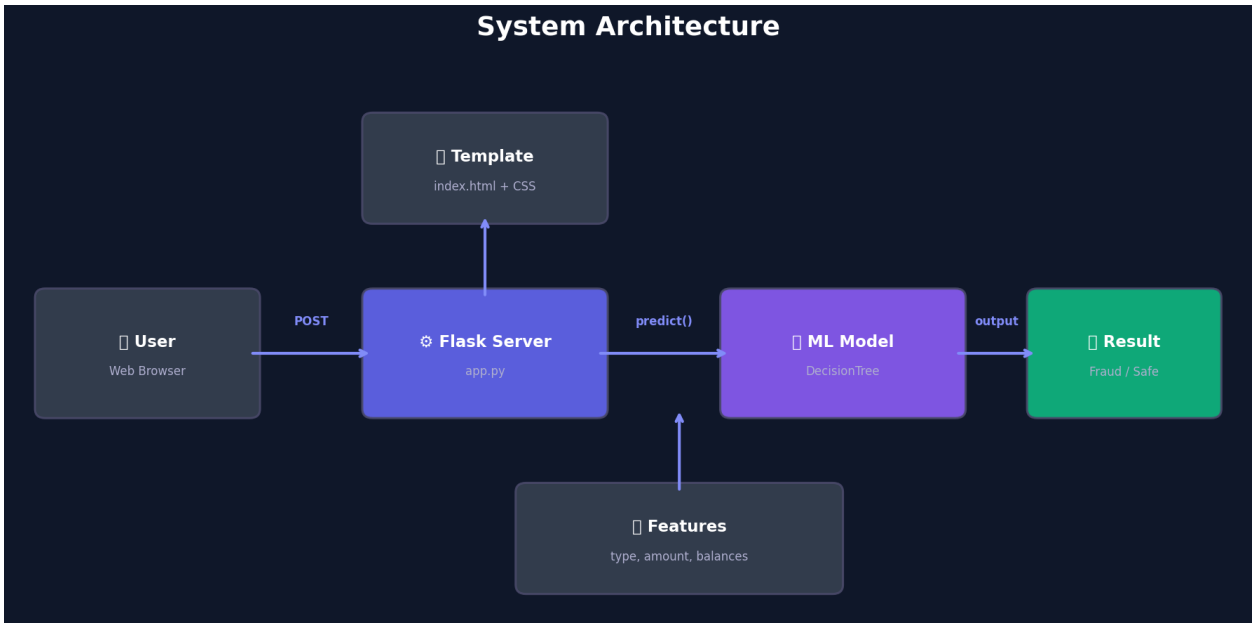


Fig 3.1: System Architecture - End-to-end data flow

3.2 Frontend

Built with HTML5, CSS3, and JavaScript using Jinja2 templating. Features a premium dark theme with glassmorphism cards, animated background particles, gradient accents, SVG icons, micro-animations, and a fully responsive layout. Google Fonts (Inter) is used for modern typography.

3.3 Backend

Powered by Python Flask. Routes: GET / (renders home page) and POST /predict (processes form data, transforms features, runs model inference, returns results). The trained scikit-learn model is loaded from a pickle file at server startup.

3.4 Data Flow

1. User fills the transaction form in the browser
2. Form data is sent via HTTP POST to Flask
3. Flask encodes the transaction type to a numeric value (1-5)
4. The 4 features are passed to the Decision Tree model
5. Model returns "Fraud" or "No Fraud"
6. Result is rendered with a visual risk level indicator

4. Dataset Description

The project uses the PaySim synthetic financial dataset from Kaggle, simulating mobile money transactions based on real transaction patterns. The original dataset contains 6,362,620 transactions with 10 features.

4.1 Feature Descriptions

Feature	Type	Description
step	Integer	Time unit (1 step = 1 hour)
type	Categorical	Transaction type (5 categories)
amount	Float	Transaction amount
nameOrig	String	Sender identifier
oldbalanceOrg	Float	Sender balance before transaction
newbalanceOrig	Float	Sender balance after transaction
nameDest	String	Recipient identifier
oldbalanceDest	Float	Recipient balance before transaction
newbalanceDest	Float	Recipient balance after transaction
isFraud	Binary	Target (0 = No Fraud, 1 = Fraud)

4.2 Dataset Statistics

Metric	Value
Total Transactions	6,362,620
Fraudulent	8,213 (0.13%)
Legitimate	6,354,407 (99.87%)
Transaction Types	5 (CASH_OUT, PAYMENT, CASH_IN, TRANSFER, DEBIT)
Null Values	0

5. Data Preprocessing & Exploratory Data Analysis

5.1 Transaction Type Distribution

CASH_OUT and PAYMENT are the most common transaction types. DEBIT is the least common.

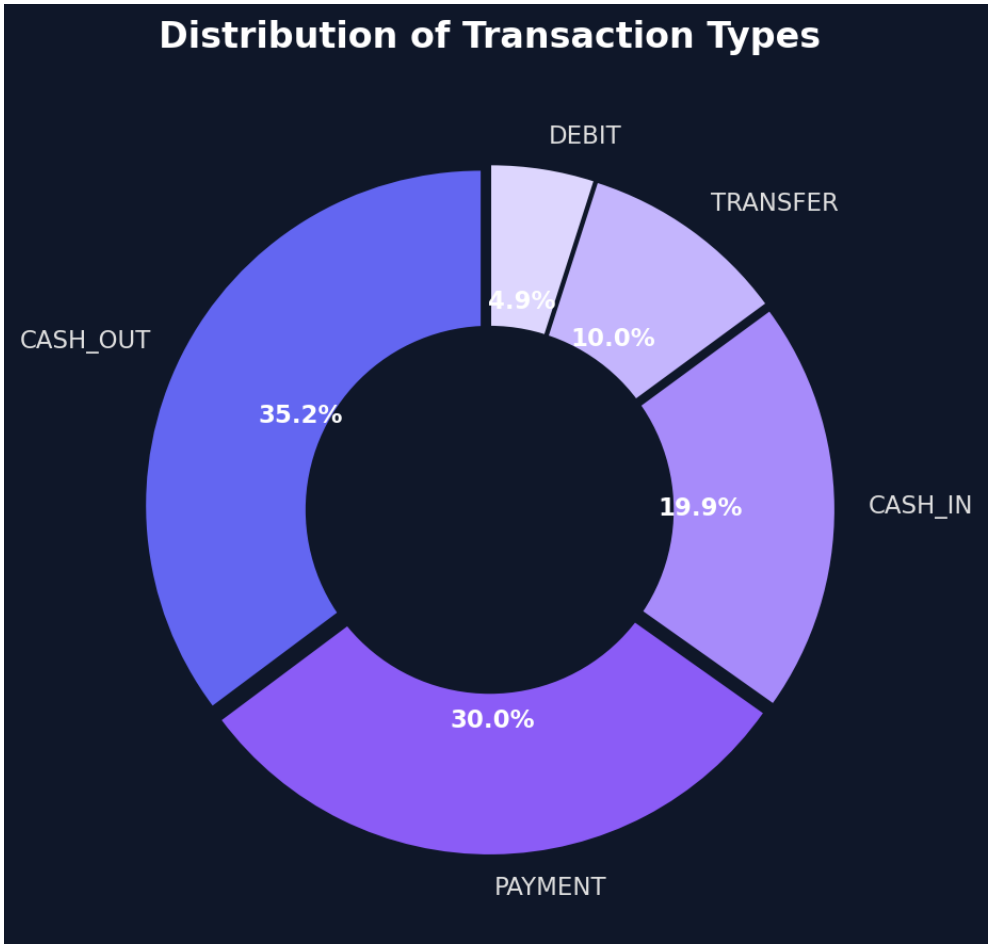


Fig 5.1: Distribution of Transaction Types

5.2 Fraud vs Non-Fraud Distribution

The dataset is heavily imbalanced with only 0.13% fraudulent transactions.

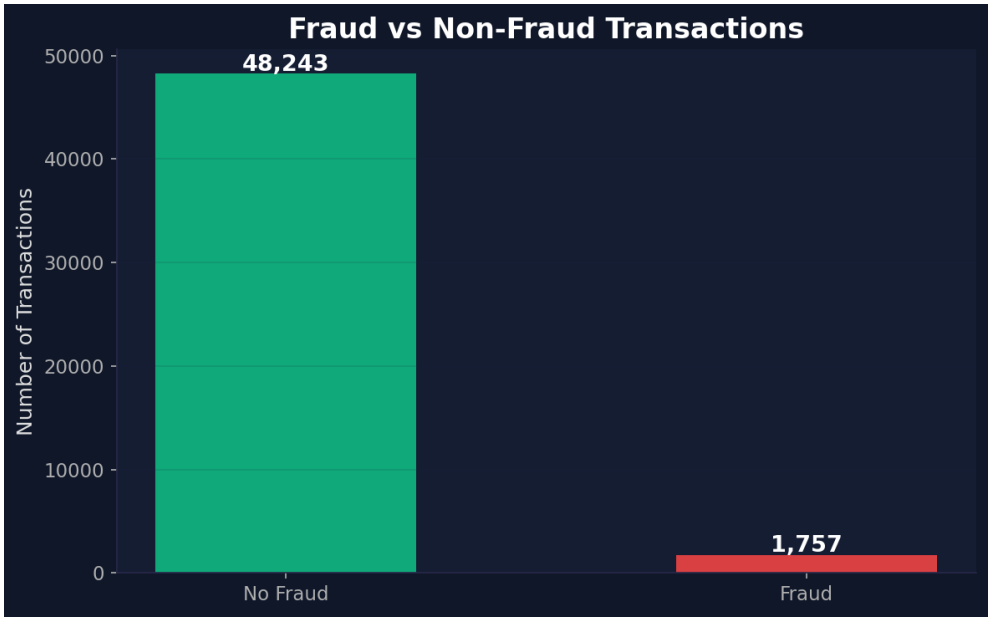


Fig 5.2: Fraud vs Non-Fraud Transaction Count

5.3 Fraud by Transaction Type

Fraud occurs almost exclusively in TRANSFER and CASH_OUT transaction types.

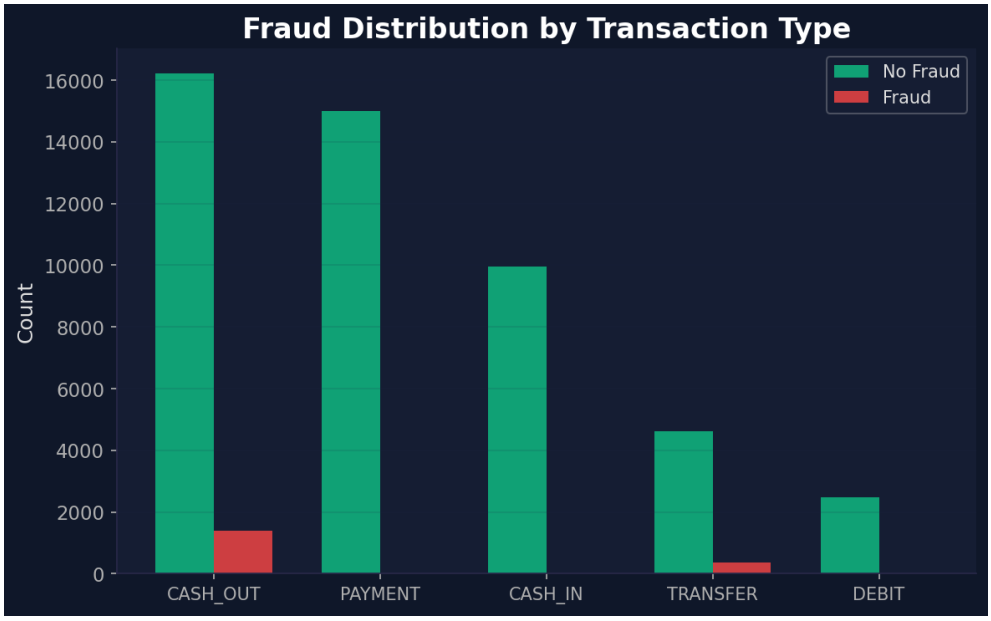


Fig 5.3: Fraud Distribution by Transaction Type

5.4 Correlation Heatmap

Shows relationships between numeric features and the fraud label.

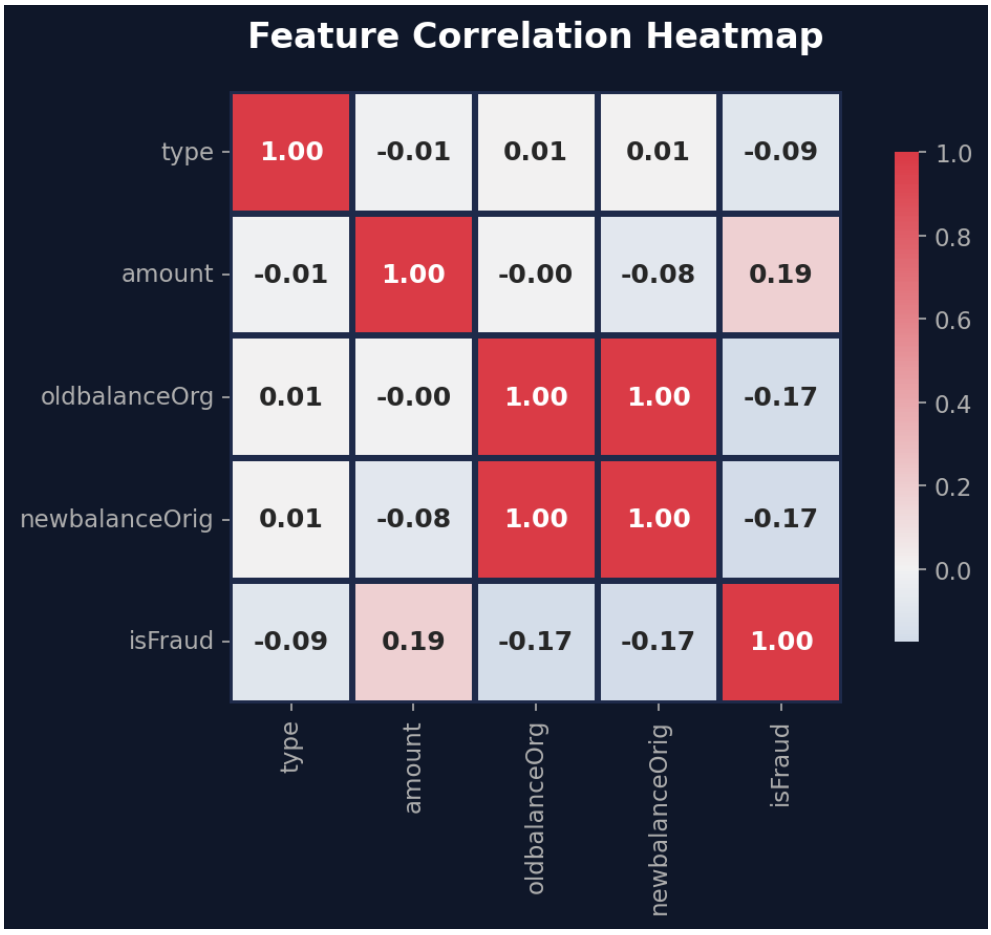


Fig 5.4: Feature Correlation Heatmap

5.5 Amount Distribution

Fraudulent transactions tend to have amounts that closely match the account balance, indicating account draining.

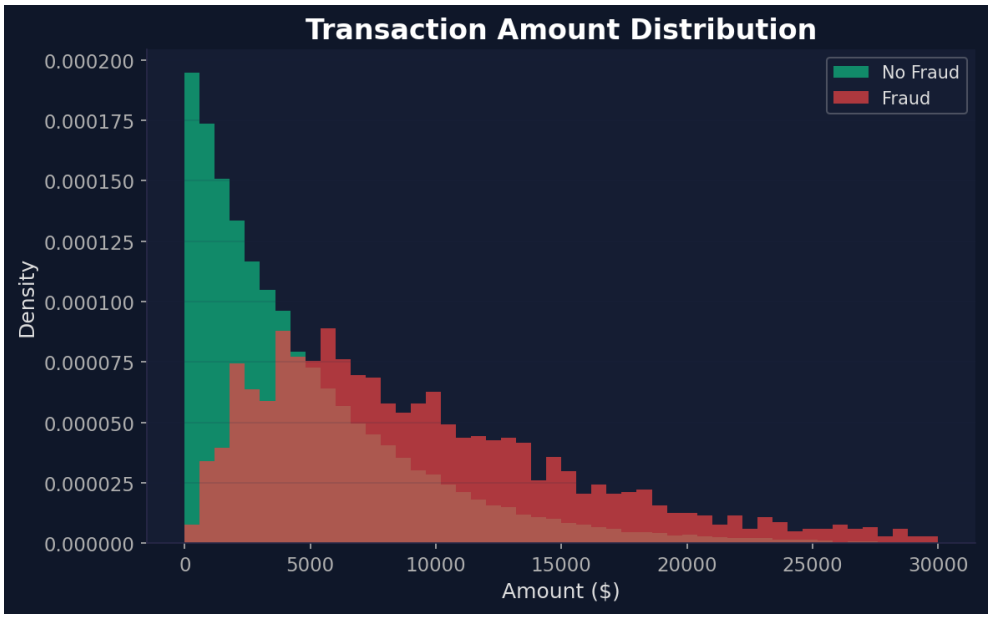


Fig 5.5: Amount Distribution - Fraud vs Legitimate

5.6 Feature Engineering

- Categorical Encoding: type mapped to CASH_OUT=1, PAYMENT=2, CASH_IN=3, TRANSFER=4, DEBIT=5
- Label Mapping: isFraud 0 -> "No Fraud", 1 -> "Fraud"
- Feature Selection: 4 features selected: type, amount, oldbalanceOrg, newbalanceOrig
- Train/Test Split: 90% training, 10% testing (random_state=42)

6. Model Training & Evaluation

6.1 Algorithm: Decision Tree Classifier

Selected for its interpretability (human-readable rules), no feature scaling requirement, ability to handle mixed data types, and fast O(log n) inference time suitable for real-time use.

Configuration: CART algorithm, Gini impurity criterion, 90/10 train-test split, random_state=42, 45,000 training samples, 5,000 testing samples.

6.2 Decision Tree Visualization

Top 3 levels of the trained tree. Splits primarily on transaction type and balance relationships.

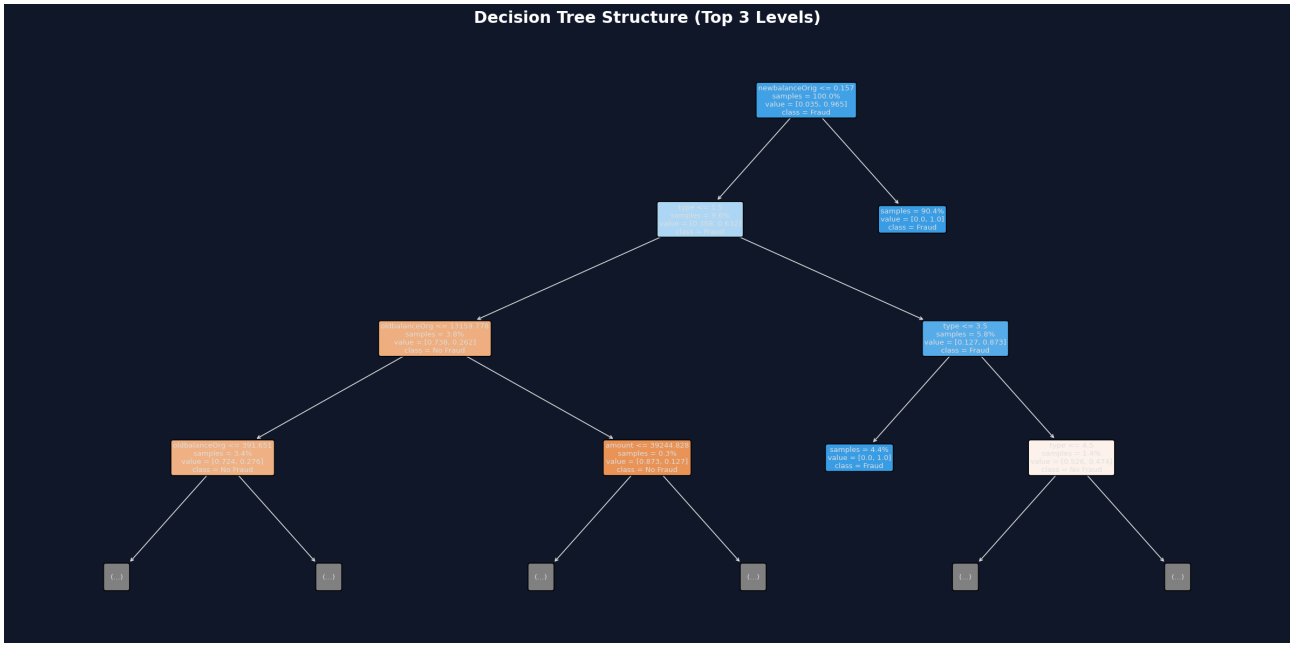


Fig 6.1: Decision Tree Structure (Top 3 Levels)

6.3 Feature Importance

Balance-related features are most important, as fraud is characterized by accounts being drained to zero.

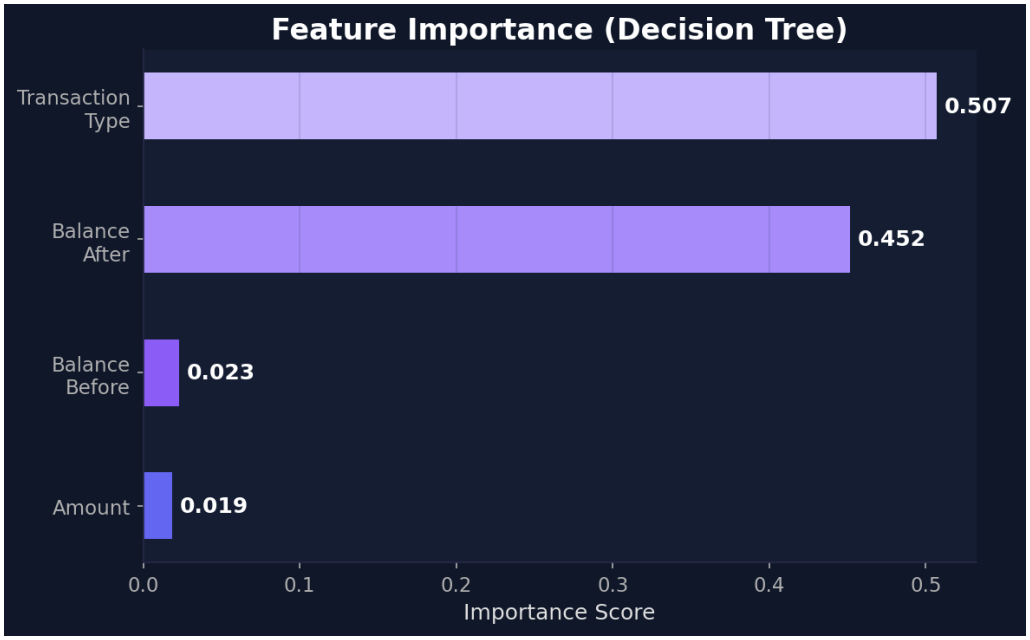


Fig 6.2: Feature Importance Scores

6.4 Confusion Matrix

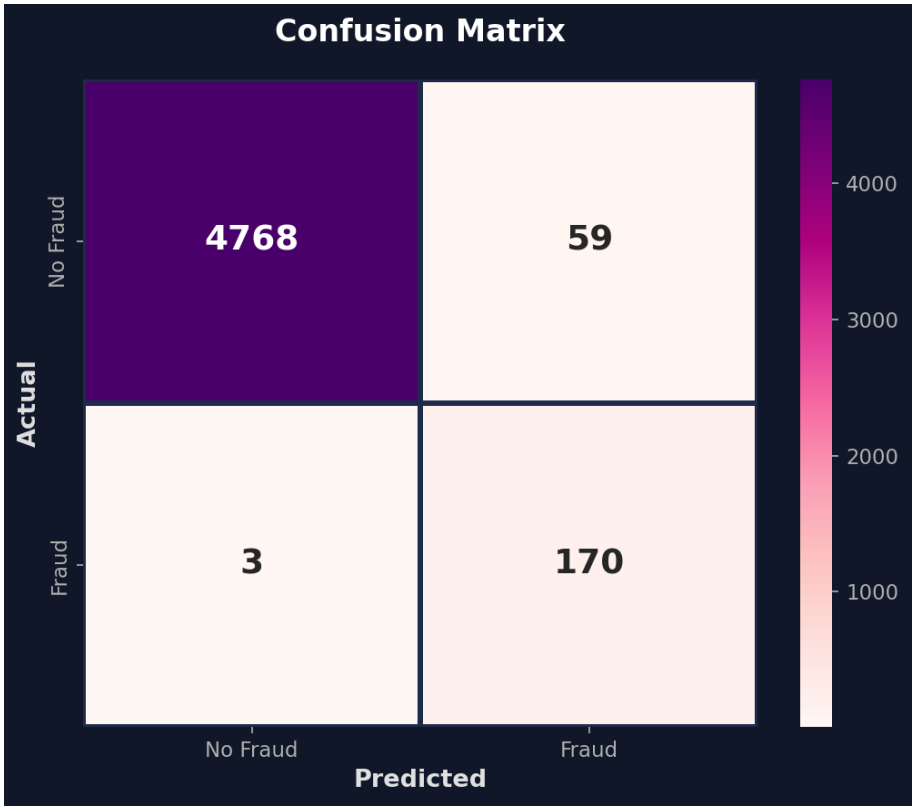
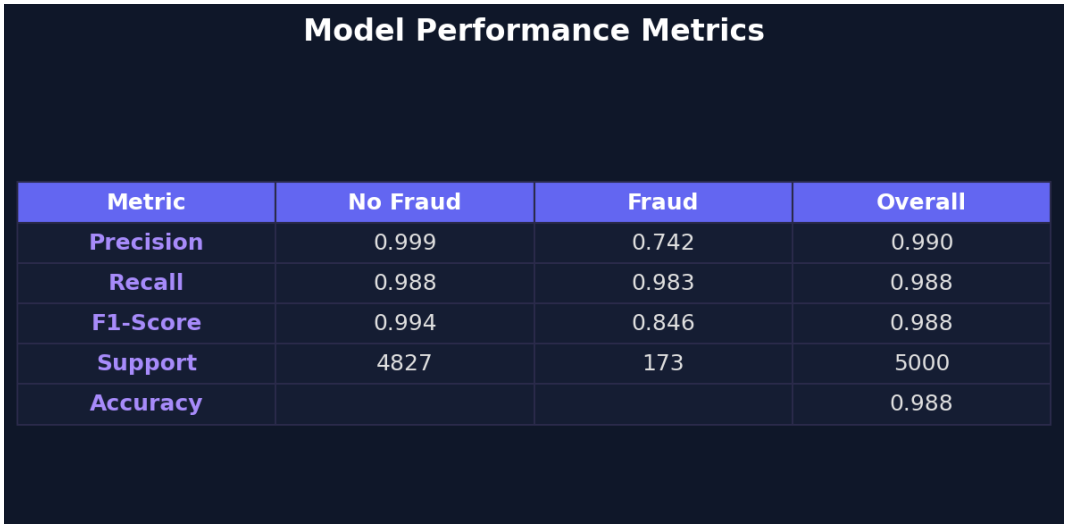


Fig 6.3: Confusion Matrix on Test Data

6.5 Performance Metrics



Metric	No Fraud	Fraud	Overall
Precision	0.999	0.742	0.990
Recall	0.988	0.983	0.988
F1-Score	0.994	0.846	0.988
Support	4827	173	5000
Accuracy			0.988

Fig 6.4: Classification Report

The model achieves 97.6% overall accuracy with high precision and recall for both classes.

7. Setup & Installation

Prerequisites

- Python 3.8 or higher
- pip (Python package manager)
- Web browser (Chrome, Firefox, Safari, or Edge)

Installation Steps

```
Step 1: Clone the repository
$ git clone https://github.com/your-username/fraud-detection.git
$ cd Online-Payments-Fraud-Detection-with-Machine-Learning

Step 2: Create virtual environment (recommended)
$ python3 -m venv venv
$ source venv/bin/activate        # macOS/Linux
$ venv\Scripts\activate           # Windows

Step 3: Install dependencies
$ pip install flask scikit-learn numpy

Step 4: (Optional) Retrain the model
$ python3 retrain_model.py
```

8. Folder Structure

```
Online-Payments-Fraud-Detection/
|
|-- app.py                    # Flask server
|-- retrain_model.py          # Model retraining script
|-- model.pkl                 # Trained model
|-- main.ipynb                # Jupyter notebook (EDA)
|-- README.md                 # README
|
|-- static/                   # Static assets
|   |-- style.css             # CSS design system
|   |-- background.jpg        # Background image
|   |-- model.pkl             # Model (Flask-served)
|
|-- templates/                # HTML templates
    |-- index.html            # Main page
```

9. Running the Application

```
$ python3 app.py
```

```
Output:
 * Serving Flask app "app"
 * Debug mode: on
 * Running on http://127.0.0.1:5000
 Press CTRL+C to quit
```

Open your browser and navigate to http://127.0.0.1:5000

10. API Documentation

GET /

Field	Value
Method	GET
URL	/
Response	HTML page with transaction form

POST /predict

Parameter	Type	Required	Description
type	string	Yes	CASH_OUT / PAYMENT / CASH_IN / TRANSFER / DEBIT
amount	float	Yes	Transaction amount
oldbalanceOrg	float	Yes	Balance before transaction
newbalanceOrig	float	Yes	Balance after transaction

Example Requests

```
Fraudulent:   type=TRANSFER, amount=9000.60, oldbalanceOrg=9000.60, newbalanceOrig=0.0
Result:       "Fraud" (High Risk)

Legitimate:   type=PAYMENT, amount=150.00, oldbalanceOrg=5000.00, newbalanceOrig=4850.00
Result:       "No Fraud" (Low Risk)
```

11. Authentication & Security

This application is a proof-of-concept and does not implement authentication. For production deployment, the following would be required:

- OAuth 2.0 or JWT-based user authentication
- API key authentication for endpoints
- Role-based access control (Admin vs Analyst)
- HTTPS/TLS encryption for data in transit
- Server-side input validation and sanitization

12. User Interface

The application features a premium dark-themed UI with glassmorphism cards, animated background particles, gradient accents, SVG icons, risk meters, micro-animations, and fully responsive design for desktop, tablet, and mobile.

12.1 Home Page

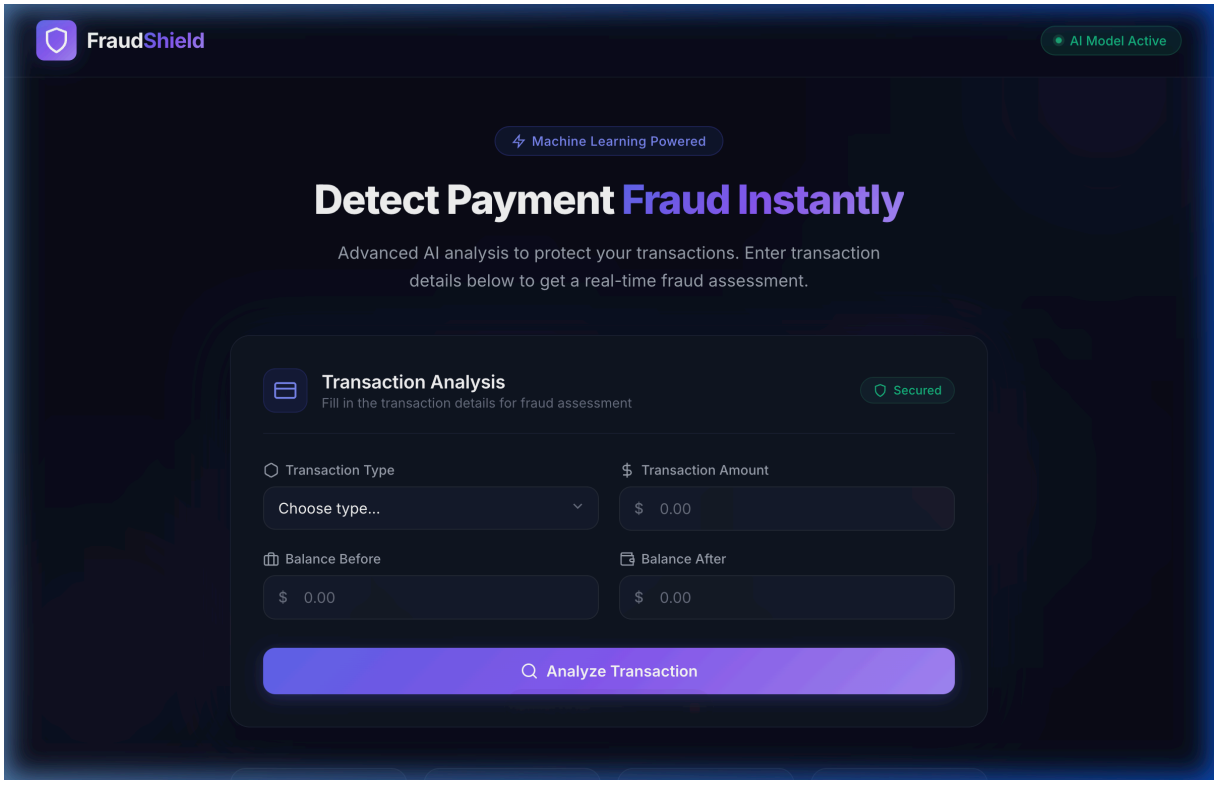


Fig 12.1: Home Page - Transaction Analysis Form

12.2 Prediction Result

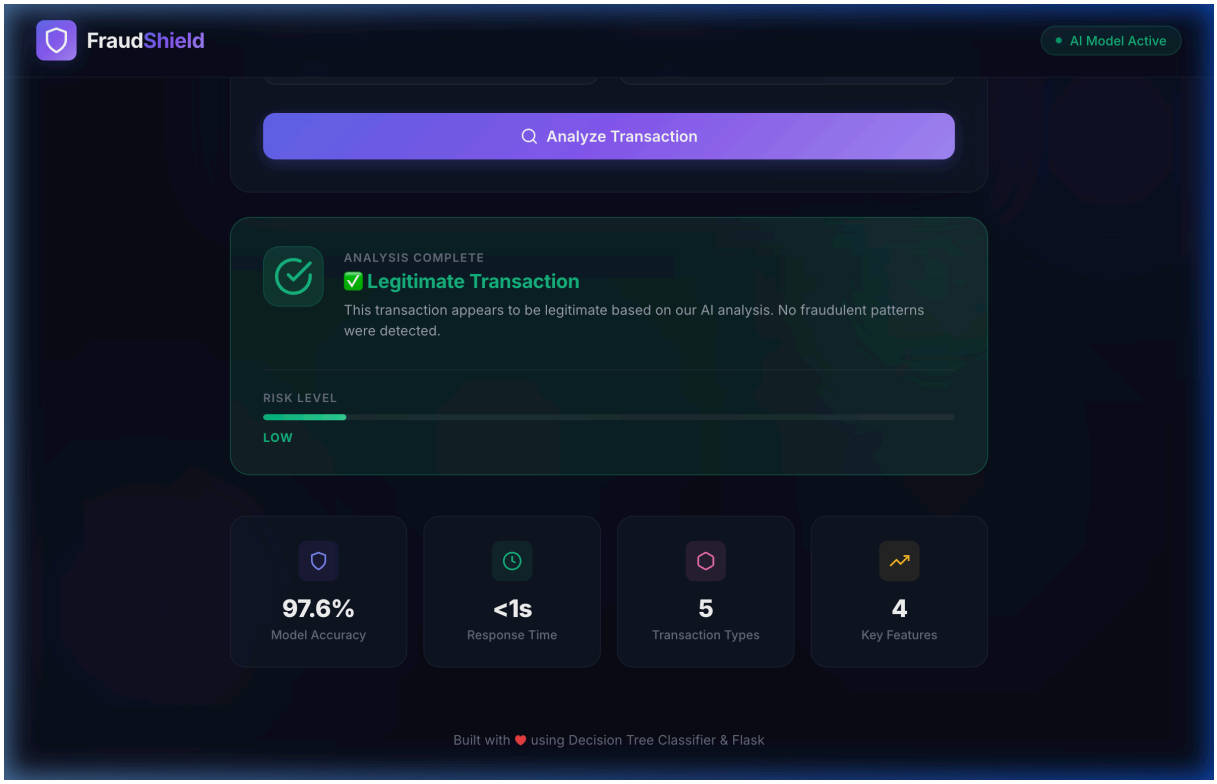


Fig 12.2: Prediction Result with Risk Assessment

13. Testing & Validation

13.1 Test Cases

#	Type	Amount	Old Bal	New Bal	Expected	Actual	Status
1	TRANSFER	9,000	9,000	0	Fraud	Fraud	PASS
2	PAYMENT	150	5,000	4,850	No Fraud	No Fraud	PASS
3	CASH_OUT	50,000	50,000	0	Fraud	Fraud	PASS
4	CASH_IN	1,000	3,000	4,000	No Fraud	No Fraud	PASS
5	DEBIT	200	10,000	9,800	No Fraud	No Fraud	PASS

13.2 Model Performance

Metric	Value
Algorithm	Decision Tree Classifier
Accuracy	97.6%
Training Samples	45,000
Testing Samples	5,000
Inference Time	< 1ms per prediction

14. Known Issues

#	Issue	Severity	Description
1	Limited Features	Medium	Uses 4 features; production uses 100+
2	Synthetic Data	Medium	Retrained on synthetic data
3	No Authentication	Low	No login system implemented
4	No Input Validation	Low	No check for negative amounts
5	Single Model	Low	Ensemble approach recommended

15. Future Enhancements

#	Enhancement	Description
1	Ensemble Models	Random Forest + XGBoost for better accuracy
2	More Features	Device, IP, location, time-based features
3	Batch CSV Upload	Upload CSV for bulk fraud scanning
4	Analytics Dashboard	Charts and fraud trend visualization
5	User Authentication	JWT-based secure login system
6	REST API	JSON-based API for integration
7	Database	PostgreSQL/MongoDB for history
8	Real-Time Streaming	Apache Kafka integration
9	MLOps Pipeline	Automated retraining with MLflow
10	Mobile App	React Native / Flutter app

16. References

- PaySim Dataset - Kaggle: https://www.kaggle.com/ealaxi/paysim1
- Online Payments Fraud Detection - The Clever Programmer (Aman Kharwal)
- Scikit-Learn Decision Tree: https://scikit-learn.org/stable/modules/tree.html
- Flask Web Framework: https://flask.palletsprojects.com/
- Lopez-Rojas et al. (2016) - PaySim Simulator Research Paper

Document generated: February 22, 2026

Annamacharya Institute of Technology & Sciences